UML (informatique)

Le **Langage de Modélisation Unifié**, de l'anglais *Unified Modeling Language* (**UML**), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé endéveloppement logicielet en conception orientée objet

L'UML est le résultat de la fusion de précédents langages de modélisation objet : Booch, OMT, OOSE. Principalement issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard adopté par l'Object Management Group (OMG).

Logo d'UML.

Sommaire

Utilisation

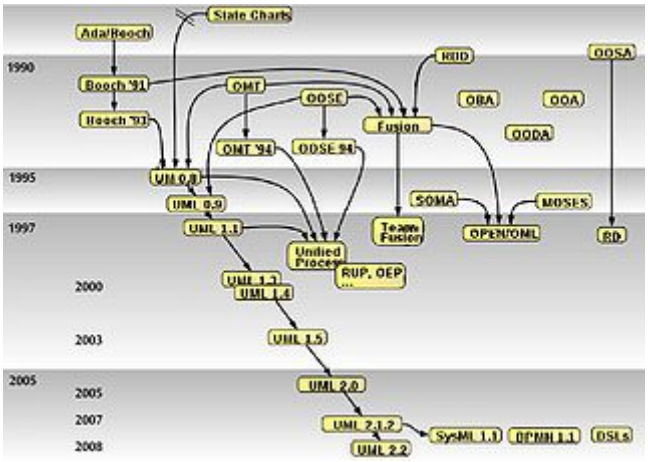
UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

- Activité d'un objet/logiciel
- Acteurs
- Processus

- Schéma de base de données
- Composants logiciels
- Réutilisation de composants

Grâce aux outils de modélisation UML, il est également possible de générer automatiquement tout ou partie du code d'une applicatio logicielle, par exemple en langageJava, à partir des divers documents réalisés.

Histoire



Historique d'UML.

Historique

Date	Description
Au début des années 1980	Les objets commencent à quitter les laboratoires de recherche et à faire leurs premiers pas dans le monde réel; entre autres, le langage de programmation Smalltalk, stabilisé, devient une plate-forme utilisable et le C++ voit le jour. Les méthodes objets commencent à émerger pour remplacer les méthodes structurée et fonctionnelles, trop liés à la machine.
1989 à 1994	Le nombre de méthodes orientées objet passe de dix (10) à plus de cinquante (50); toutes ces méthodes ont de nombreux points communs (objets, méthodes, paramètres, etc.). Ces méthodes s'orientant sur l'abstraction des composants matériels, se basent sur des notions de classe, d'association, de partition en sous-systèmes et autour de l'étude de l'interaction entre utilisateur et le système. Les principaux auteurs de ces méthodes sont James Rumbaugh, Grady Booch et Ivar Jacobson. Parmi ces méthodes, deux s'imposent : la méthode de Booch et la méthode OMT (Object Modeling Technique). Les deuxièmes versions des méthodes de Booch et OMT font leur apparition : Booch'93 et OMT-2. Ces méthodes sont assez semblables, mais Booch'93 insiste plus sur la construction tandis que OMT-2 insiste plus sur l'analyse et l'abstraction.
1989 et 1991	Publication de deux (2) ouvrages, par Sally Shlaer (en) et Steve Mellor sur l'analyse et la conception, qui débouchent sur une approche qu'ils nomment **conception récursive**
1989 à 1990	Développement à Portland, par la communauté Smalltalk, de la conception pilotée par les responsabilités et les cartes **CRC** (Class-Responsability-Collaboration).
1991 à 1996	James Rumbaugh dirige aux laboratoires de recherche de General Electric une équipe de recherche qui publie un ouvrage très apprécié sur OMT.
1991	Publication d'ouvrages, par Peter Coad et Ed. Yourdon, qui développent les approches « allégées » et « orientées prototypes ».
1992 et 1995	Publication des livres d'Ivar Jacobson fondés sur son expérience des commutateurs téléphoniques chez Ericsson. Le premier introduit le concept de cas d'utilisation (use-case).
1994 à 1996	Grady Booch effectue un travail important chez Rational Software en développant des systèmes en Ada.
1994	Le nombre important de méthodes et le fait que les différences entre-elles se réduisent, font reculer la technologie objet au point que Jim Rumbaugh et Grady Booch s'unissent afin d'unifier leur travaux. Ils proposent une « méthode unifiée ».
1994	Les livres de Jim Odell, écrits avec James Martin, se fondent sur sa longue expérience des systèmes d'information et du génie logiciel et sont, parmi tous ces ouvrages, les plus conceptuels.
1994-10	Début des travaux de la méthode unifiée (*unified method* (**UM**)). James Rumbaugh rejoint Grady Booch chez Rational Software.
1995	Ivar Jacobson, créateur des *use cases*, rejoint Jim Rumbaugh et Grady Booch.
1995	Les auteurs de la méthode unifiée (**UM**) publient le document intitulé **Unified Method V0.8**
1995-10	Ivar Jacobson arrive chez Rational Software.
1995-10	UML 0.8 inclut **OOD**/Booch '93 de Grady Booch et OMT de Jim Rumbaugh.
1996	Publication d'une nouvelle révision du document, **Unified Method V0.9**, suite aux commentaires des utilisateurs.

	La révision 0.9.1 est la version la plus aboutie de la méthode unifiée (réorientation de la portée de l'effort d'unification). La méthode change de nom et se transforme en UML (Unified Modeling Language for Object-Oriented Development). Un consortium de grandes entreprises se crée (Microsoft, IBM, Oracle, etc.) qui permettra de faire passer la méthode à sa version 1.0.
1996-06	UML 0.9 inclut OOSE d'Ivar Jacobson.
1996-10	UML 0.91
1997-01	Normalisation de UML 1.0 par l'Object Management Group (OMG).
1997-08	Proposition des spécifications d'UML 1.1 à l'OMG par un groupe de travail d'analystes et de concepteurs dirigé par Cris Kobryn et administré par Ed Eykholt.
1997-11-14	Adoption des spécifications d'UML 1.1 par l'OMG[1]. Différentes améliorations continuent d'être apportées au standard UML, donnant naissance à 4 révisions : UML 1.2, 1.3, 1.4, 1.5. UML 1.5 est la dernière révision avant le passage à la version UML 2.0. Les standards UML 1.x, encore largement influencés par la notation OMT, est décrié comme manquant d'intégration sémantique.
1998-06	Adoption de UML 1.2 par l'OMG.
1998-10	Adoption de UML 1.3 par l'OMG.
2000-03	Publication de la spécification UML 1.3 complète.
2001-09	UML 1.4.
2003-03-06	UML 1.5 (recommandations)
2003-08	UML 2.0 Superstructure Specification (recommandation)
2003-09-01	UML 2.0 Diagram Interchange Specification (recommandation)
2003-10-14	UML 2.0 OCL Specification
2003-12	UML 2.0 (recommandation)
2005-07	Adoption de UML 2.0 par l'OMG.
2006-04-04	UML 2.0 Diagram Interchange Specification
2006-06-01	*deployment view*)
2006-10-06	UML 2.1.1 - XMI file
2007-02-06	UML 2.1.1 Infrastructure Specification
2007-02-03	UML 2.1.1 Superstructure Specification
2007	UML 1.4.2 devient une spécification ISO (ISO/IEC 19501).

2007-11	Diffusion de UML 2.1.2 par l'OMG.
	L'OMG travaille à UML 2.2.
2013-09	Diffusion par l'OMG d'UML 2.5 bêta 2.

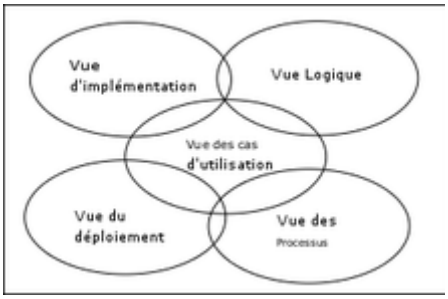
Formalisme

UML 2.3 propose 14 types de diagrammes (25 en UML 1.3). UML n'étant pas une méthode, leur utilisation est laissée à l'appréciation de chacun, même si le diagramme de classes est généralement considéré comme l'élément central d'UML ; des méthodologies, telles que l'UnifiedProcess, axent l'analyse en tout premier lieu sur les diagrammes de cas d'utilisation (use case). De même, on peut se contenter de modéliser seulement partiellement un système, par exemple certaines parties critiques.

- UML se décompose en plusieurs parties :
  - Les *vues* : ce sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.
  - Les *diagrammes* : ce sont des ensembles d'éléments graphiques. Ils décrivent le contenu des vues, qui sont des notions abstraites. Ils peuvent faire partie de plusieurs vues.
  - Les *modèles d'élément* : ce sont les éléments graphiques des diagrammes.

Vues

Une façon de mettre en œuvre UML est de considérer différentes *vues* qui peuvent se superposer pour collaborer à la définition du système :



Vues d'UML.

- *Vue des cas d'utilisation* (*use-case view*) : c'est la description du modèle vu par les acteurs du système. Elle correspond aux besoins attendus par chaque acteur (c'est le quoi et le qui).
- *Vue logique* (*logical view*) : c'est la définition du système vu de l'intérieur. Elle explique comment peuvent être satisfaits les besoins des acteurs (c'est le comment).
- *Vue d'implémentation* (*implementation view*) : cette vue définit les dépendances entre les modules.
- *Vue des processus* (*process view*) : c'est la vue temporelle et technique, qui met en œuvre les notions de tâches concurrentes, stimuli, contrôle, synchronisation…
- *Vue de déploiement* (*deployment view*) : cette vue décrit la position géographique et l'architecture physique de chaque élément du système (c'est le où).

Le pourquoi n'est pas défini dans UML.

Diagrammes

Les *diagrammes* sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe quatorze depuis UML 2.3.


La hiérarchie des diagrammes UML 2.0 sous forme d'un diagramme de classes.

Diagrammes de structure ou diagrammes statiques

Les *diagrammes de structure* (*structure diagrams*) ou *diagrammes statiques* (*static diagrams*) rassemblent :

- *Diagramme de classes* (*class diagram*) : représentation des classes intervenant dans le système.
- *Diagramme d'objets* (*object diagram*) : représentation des instances de classes (objets) utilisées dans le système.

- *Diagramme de composants* (*component diagram*) : représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données…)
- *Diagramme de déploiement* (*deployment diagram*) : représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage…) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.
- *Diagramme des paquets* (*package diagram*) : représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.
- *Diagramme de structure composite* (*composite structure diagram*) : représentation sous forme de boîte blanche les relations entre composants d'une classe (depuis UML 2.x).
- *Diagramme de profils* (*profile diagram*) : spécialisation et personnalisation pour un domaine particulier d'un meta-modèle de référence d'UML (depuis UML 2.2).

Diagrammes de comportement

Les *diagrammes de comportement* (*behavior diagrams*) rassemblent :

- *Diagramme des cas d'utilisation* (*use-case diagram*) : représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.
- *Diagramme états-transitions* (*state machine diagram*) : représentation sous forme de machine à états finis le comportement du système ou de ses composants.
- *Diagramme d'activité* (*activity diagram*) : représentation sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants.

Diagrammes d'interaction ou diagrammes dynamiques

Les *diagrammes d'interaction* (*interaction diagrams*) ou *diagrammes dynamiques* (*dynamic diagrams*) rassemblent :

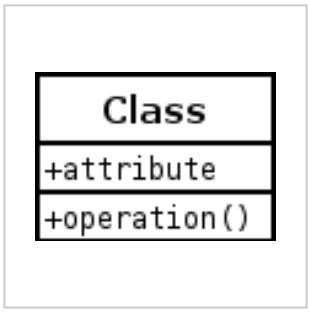
- *Diagramme de séquence* (*sequence diagram*) : représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.
- *Diagramme de communication* (*communication diagram*) : représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x).
- *Diagramme global d'interaction* (*interaction overview diagram*) : représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité) (depuis UML 2.x).
- *Diagramme de temps* (*timing diagram*) : représentation des variations d'une donnée au cours du temps (depuis UML 2.3).

Modèles d'éléments

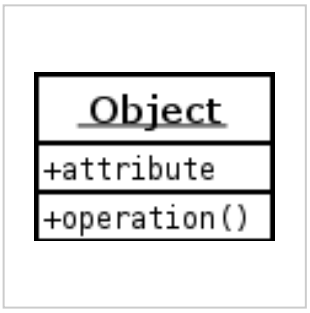
- Un *stéréotype* est une marque de généralisation notée par des guillemets, cela montre que l'objet est une variété d'un modèle.
- Un *classeur* est une annotation qui permet de regrouper des unités ayant le même comportement ou structure. Un classeur se représente par un rectangle conteneur en traits pleins.
- Un *paquet* regroupe des diagrammes ou des unités.
- Chaque classe ou objet se définit précisément avec le signe « :: ». Ainsi l'identification d'une classe X en dehors de son paquet ou de son classeur sera définie par « Paquet A::Classeur B::Classe X ».

Modèles d'éléments de type commun

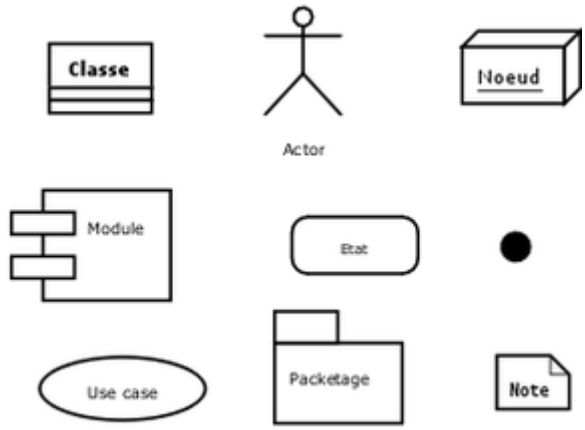
Symbolique des modèles d'éléments :



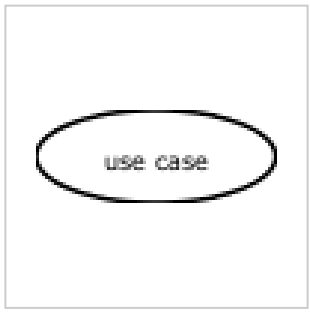
Classe (*class*).



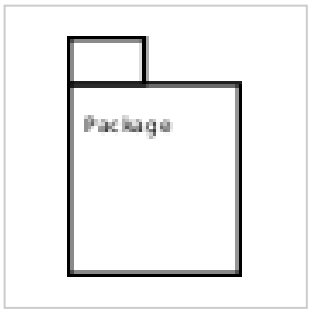
Objet (*object*).



Modèles d'éléments UML.



Cas d'utilisation (*use case*).

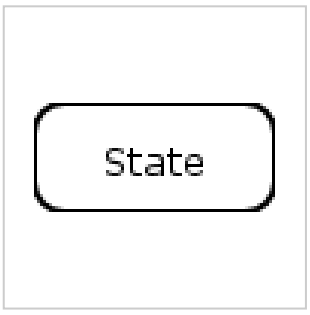


Paquet (*package*).

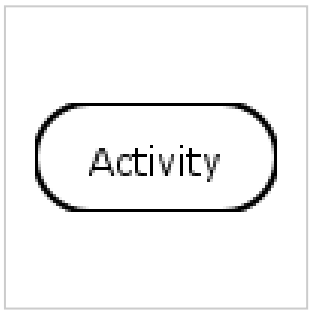
Nœud (*node*).

- Fourche (*fork*).

Acteur (*actor*).



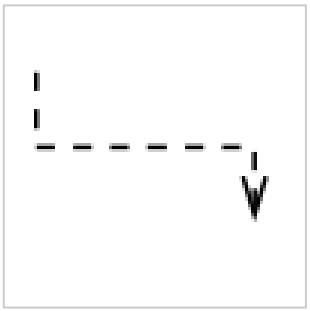
État (*state*).



Activité (*activity*).

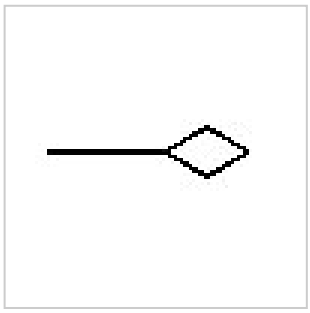
- État initial (*initial state*).
- État final (*final state*).
- Interface (*interface*).
  - O←--- sens du flux de l'interface.
  - O)----- est un raccourci pour la superposition de ---→O et O←---.

Modèles d'éléments de type relation

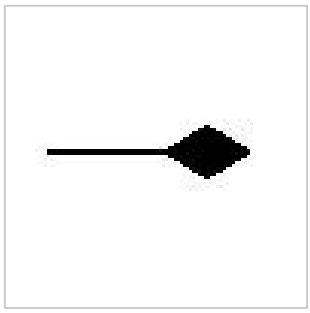


Dépendance (*dependency*).

- Généralisation (*generalisation*).
- Association (*association*).



Agrégation (*aggregation*).



Composition (*composition*).

- Réalisation.
- Utilisation.

Autres modèles d'éléments

- Les stéréotypes peuvent dépendre du langage que l'on souhaite utiliser
- Les archétypes.
- Les profils.

Normalisation et certification

UML n'est pas une norme en droit mais un simple standard « industriel » (ou norme de fait), parce que promu par l'OMG (novembre 1997) au même titre que CORBA et en raison de son succès. Depuis juillet 2005, la première version 2.x de UML est validée par l'OMG.

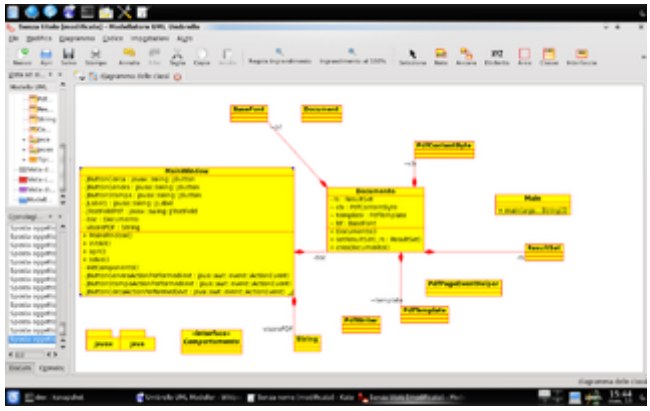
Par ailleurs, depuis 2003, l'OMG a mis en place un programme de certification à la pratique et la connaissance d'UML OCUP (*OMG Certified UML Professional*), qui recouvre trois niveaux successifs de maîtrise.

Exemple de séquence de création des diagrammes

Diagramme	Étape du cycle en V
1. Diagramme de cas d'utilisation	Spécification, cahier des charges
2. Diagramme de séquence	
3. Diagramme d'activité (processus métiers)	
4. Diagramme d'activité (cinématique et/ou processus applicatifs)	
5. Diagramme de classe	Conception architecturale
6. Diagramme d'objets	
7. Diagramme de communication	
8. Diagramme de déploiement	
9. Diagramme de composants	

Logiciels de modélisation UML

Il existe de nombreux logiciels de modélisation UML. Malheureusement, aucun d'entre eux ne respecte strictement chacune des versions de UML, particulièrement UML 2. Beaucoup de ces logiciels introduisent des notations particulières non conformes et très peu supportent les différents types de diagrammes définis par le standard. En revanche, de nombreux logiciels incluent des outils de génération de squelette de code, particulièrement à partir du diagramme de classes, qui est celui qui se prête le mieux à une telle automatisation.



Umbrello, atelier UML de KDE.

Notes et références

2.5 Bêta 2 http://www.omg.org/spec/UML/2.5/Beta2/

1. « UML Specification version 1.1 (OMG document ad/97-08-11) » (http://www.omg.org/cgi-bin/doc?ad/97-08-11), Omg.org (consulté le 22 septembre 2011)
2. UML 2.4.1 (http://www.omg.org/spec/UML/2.4.1)
3. OCUP (http://www.omg.org/uml-certification)

(en) OMG, « OMG Unified Modeling LanguageTM (OMG UML),Superstructure », OMG, aout 2011 (consulté le 20 novembre 2011)

(en) OMG, « OMG Unified Modeling LanguageTM (OMG UML), Infrastructure », OMG, aout 2011 (consulté le 20 novembre 2011)

Voir aussi

Bibliographie

- Grady Booch, James Rumbaugh, Ivar Jacobson, *Le guide de l'utilisateur UML*, 2000 (ISBN 2-212-09103-6)
- Laurent Audibert, *UML 2, De l'apprentissage à la pratique (cours et exercices)*, Ellipses, 2009 (ISBN 978-2729852696)
- Franck Barbier, *UML 2 et MDE, Ingénierie des modèles avec études de cas*, 2009 (ISBN 978-2-10-049526-9)
- Craig Larman, *UML 2 et les design patterns, Analyse et conception orientées objet et développement itératif* (3e édition), Pearson Education, 2005 (ISBN 2-7440-7090-4)
- Martin Fowler *et al.*, *UML 2.0, Initiation aux aspects essentiels de la notation*, 2004 (ISBN 2-7440-1713-2)

Sur les autres projets Wikimedia :

UML (informatique), sur Wikimedia Commons

Modélisation UML, sur Wikiversity

UML (informatique), sur Wikibooks

- Pascal Roques, *UML 2, Modéliser une application Web*, Eyrolles, 2007 (ISBN 2-212-12136-9)
- Pascal Roques, *UML 2 par la pratique, Études de cas et exercices corrigés*, Eyrolles, 2006 (ISBN 2-212-12014-1)
- Jim Conallen, *Concevoir des applications web avec UML*, Eyrolles, 2000, 288 p. (ISBN 978-2212091724)

Articles connexes

- Enterprise Architect
- Unified Process
- Ingénierie dirigée par les modèles
- MDA
- MTL **(en)**
- ATL
- OCL
- Transformation de modèles **(en)**
- Modeling and Analysis of Real Time and Embedded systems
- Merise (informatique)
- Acceleo
- BOUML
- Modelio
- Open ModelSphere
- QVT
- SysML
- StarUML
- WebML
- Lucidchart service d'informatique en nuage
- Les Processus d'affaires dont le Business process model and notation

Liens externes

- **(en)** UML.org
- **(en)** OMG (Object Management Group)
- **(en)** UML liste des outils
- **(en)** Profil UML standardisé par l'ITU-T basé sur le Specification and Description Language

---